(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

PCT

(43) International Publication Date
23 February 2006 (23.02.2006)

(10) International Publication Number
WO 2006/020343 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/US2005/025975

(22) International Filing Date: 21 July 2005 (21.07.2005)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/916,547 11 August 2004 (11.08.2004) US

(71) Applicant *(for all designated States except US)*: **ORACLE INTERNATIONAL CORPORATION** [US/US]; 500 Oracle Parkway, Redwood Shores, CA 94065 (US).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **DAS, Souripriya** [US/US]; 18 Sky Country Drive, Nashua, NH 03062 (US). **CHONG, Eugene, Inseok** [US/US]; 9 Dunbar Way, Concord, MA 01742 (US). **EADON, George** [US/US]; 20 Chadwick Circle, Apt. J, Nashua, NH 03062 (US). **SRINIVASAN, Jagannathan** [IN/US]; 1 Hampshire Drive, Apt. F, Nashua, NH 03063 (US).

(74) Agent: **NELSON, Gordon, E.**; 57 Central Street, P.O. Box 782, Rowley, MA 01969 (US).

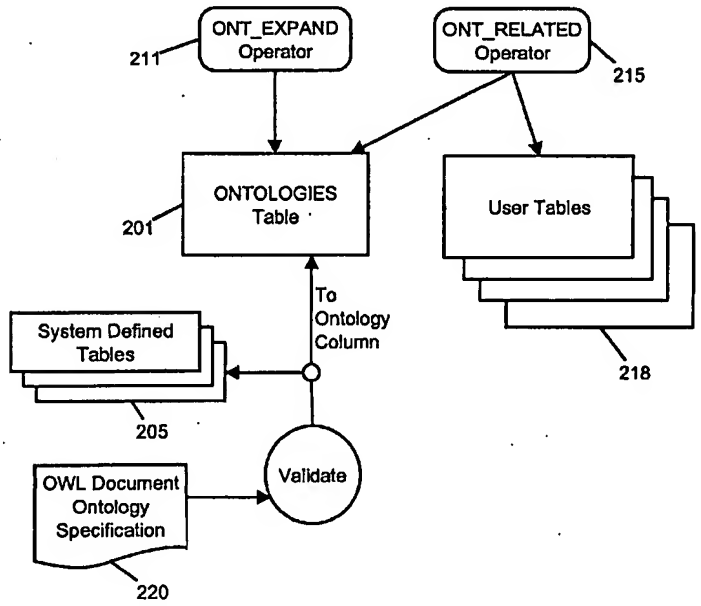
(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:
— *with international search report*

[Continued on next page]

(54) Title: SYSTEM FOR ONTOLOGY-BASED SEMANTIC MATCHING IN A RELATIONAL DATABASE SYSTEM



(57) Abstract: The method for processing data in a relational database wherein ontology data that specifies terms and relationships between pairs of said terms expressed in an OWL document is stored in the database, database queries that include a semantic matching operator are formed which identify the ontology data and further specify a stated relationship between two input terms, and the query is executed to invoke the semantic matching operator to determine if the two input terms are related by the stated relationship by consulting said ontology data.

WO 2006/020343 A1

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Patent Specification

Title: System for ontology-based semantic matching in a relational database system

Invented by

Souripriya Das
18 Sky Country Drive
Nashua, NH 03062 US
A citizen of the US

Eugene Inseok Chong
9 Dunbar Way
Concord, MA 01742 US
A citizen of the US

George Eadon
20 Chadwick Circle, Apt. J
Nashua, NH 03062 US
A citizen of the US

Jagannathan Srinivasan
1 Hampshire Dr, Apt F
Nashua, NH 03063 US
A citizen of India

Attorney Docket No. oracle01.049

Gordon E. Nelson, Patent Attorney #30,093
Patent Attorney
P.O. Box 782, 57 Central Street, Rowley, MA 01969
Phone: 978-948-7632 Fax: 866-723-0359
USPTO Customer No. 501315

SYSTEM FOR ONTOLOGY-BASED SEMANTIC MATCHING IN A RELATIONAL DATABASE SYSTEM

Field of the invention

[0001] This invention relates to methods and apparatus for storing and processing ontology data within a relational database management system (RDBMS).

Background of the invention

[0002] A single term often has different meanings in different contexts: the term "mouse" may refer to an animal in one context or to a computer input device in another. Different terms can mean the same thing, like the terms "TV" and "television." And terms may be related to one another in special ways; for example, a "poodle" is always a "dog" but a "dog" is not always a "poodle".

[0003] Humans learn to cope with the ambiguity of language by understanding the context in which terms are used. Computers can be programmed to do the same thing by consulting data structures called "ontologies" that represent terms and their interrelationships.

[0004] Data processing operations commonly need to match one term against another. Because a single term can have different meanings in different contexts, and different terms can mean the same thing, simply testing two values for equality often isn't sufficient. Consider, for example, a computerized restaurant guide application that recommends restaurants to a user based on her preferences. Such an application might employ a database table called "served_food" that identifies each restaurant by its ID number "R_id" in one column and by the kind of food it serves in a second column called "Cuisine." In the absence of semantic matching, if the user wished to identify restaurants serving Latin American cuisine, a conventional database application would most likely resort to a syntactic matching query using an equality operator as illustrated by the following SQL SELECT statement:

[0005] SELECT * FROM served_food WHERE cuisine = 'Latin American';

[0006] But this query would not identify restaurants listed as serving "Mexican," "Spanish," or "Portuguese" cuisine, since none of those terms identically match the term "Latin American" used in the query.

[0007] More meaningful results could be obtained by performing semantic matching which would take into account the meaning of terms. To do that, the matching process could consult an ontology like the one shown graphically in Fig. 1 which shows that the term 'Latin American" encompasses the more specific cuisine types identified by the terms "Mexican," "Spanish" and "Portuguese."

[0008] The equality operation commonly used in a conventional database system only allows for matching based on the structure of the data type and doesn't take into account the semantics pertaining to a specific domain. Semantic meaning can be specified by one or more ontologies associated with the domain. In recent years, mechanisms for handling ontologies have received wide attention in the context of semantic web. See, for example, "The Semantic Web" by T. Berners-Lee, J. Hendler and O. Lassila in Scientific American, May, 2001. Tools for building and using ontologies have become available and include, for example: (1) OntologyBuilder and OntologyServer from VerticalNet described by A. Das, W. Wu, and D. McGuinness in "Industrial Strength Ontology Management," The Emerging Semantic Web, IOS Press, 2002, and (2) KAON described by B. Motik, A. Maedche, and R. Volz in "A Conceptual Modeling Approach for Semantics-Driven Enterprise Applications," Proceedings of the 2002 Confederated Int. Conferences DOA/CoopIS/ODBASE, 2002. These tools permit ontologies to be stored in a relational database, and provide a procedural API (application program interface) for accessing and manipulating the ontologies. To incorporate ontology- based semantic matching into an application, however a user needs to make use of the provided APIs to first query the ontology and then combine the results from the API with queries on database tables, a process that is burdensome to the user and requires additional processing.

[0009] The formal specification of an ontology facilitates building applications by separating the knowledge about the target domain from the rest of the application code. This separation substantially simplifies the application code, makes it easier to share the knowledge represented by the ontology among multiple applications, and allows that knowledge to be expanded or corrected without requiring changes to the application.

[0010] Relational database systems that are in widespread use must utilize ontologies to provide improved results. To achieve that, however, the existing capabilities of the RDBMS must be expanded to support ontology-based semantic matching in a relational database

management system (RDBMS), and these enhanced capabilities should be made available in ways that are consistent with existing practices that are already familiar to database users.

Summary of the invention

[0011] The preferred embodiment of the present invention extends the capabilities of an existing relational database management system by introducing a set of new SQL operators, here named ONT_RELATED, ONT_EXPAND, ONT_DISTANCE, and ONT_PATH, to perform ontology-based semantic matching. These operators allow database users to reference ontology data directly using SQL statements can combine these semantic matching operators with other conventional SQL operations such as joins to make use of the full expressive power of SQL while performing semantic based matching. The semantic matching operators contemplated by the present invention make new, efficient ontology-based applications easy to develop, and make it possible to easily enhance existing RDBMS applications to obtain the benefit of semantic matching.

[0012] The ONT_RELATED operator performs ontology-based semantic matching and is expressed within an SQL statement using an expression of the form: "ONT_RELATED(term1, reltype, term2, ontology)."

[0013] When executed, the ONT_RELATED operator determines whether the two input terms (term1 and term2) are related by the specified input relationship type "reltype" by consulting the ontology.

[0014] Prior to executing a query containing the semantic matching operator, the specified ontology is registered with the database and mapped into system defined tables.

[0015] Two ancillary operators, ONT_DISTANCE and ONT_PATH, are employed to determine additional measures for the matched rows that are identified, namely, shortest distance and shortest path, respectively. These operators identify the terms that are most closely matched in the specified ontology. Two additional ancillary operators named ONT_DISTANCE_ALL and ONT_PATH_ALL return distance measures and path respectively for all matching terms identified in the ontology.

[0016] An operator named ONT_EXPAND expressed in the form ONT_EXPAND(term1, reltype, term2, ontology) is employed to directly access the ontology data. This operator computes the transitive closure for (term1, term2) based on the specified relationship (reltype) of the specified ontology.

[0017] The term1, reltype, and term2 can have either a specific input value or NULL value. The NULL means all possible values. For example, ONT_EXPAND(NULL, 'IS_A', 'Vehicle') will generate all terms that are related by 'IS_A' relationship to the term 'Vehicle'.

Brief description of the drawings

[0018] In the detailed description which follows, frequent reference will be made to the attached drawings, in which:

[0019] Fig. 1 is graph depicting a illustrative ontology that defines hierarchical relationships between terms used to describe food served by restaurants; and

[0020] Fig. 2 is a block diagram illustrating the principle data structures used to implement the preferred embodiment of the invention;

[0021] Fig. 3 is a diagram illustrating the addition of an EQV relationship; and

[0022] Fig. 4 is a diagram illustrating the manner in which indexing is used to speed term matching operations.

Detailed description

[0023] **1. Introduction**

[0024] The present invention employs a set of SQL (Structured Query Language) operators to perform ontology-based semantic matching on data stored in a relational database management system (RDBMS). These SQL operators preferably take the form of extensions to the pre-existing SQL syntax employed by the database and may be implemented with the database extensibility capabilities (namely, the ability to define user-defined operators, user-defined indexing schemes, and table functions) typically available in a robust database system. [0025] The specific embodiment of the invention described below has been implemented on top of the existing SQL syntax used in the Oracle family of databases. Detailed information on the Oracle SQL language and its syntax can be found in

the Oracle 8i SQL Reference available from Oracle Corporation. This reference contains a complete description of the Structured Query Language (SQL) used to manage information in an Oracle database. Oracle SQL is a superset of the American National Standards Institute (ANSI) and the International Standards Organization (ISO) SQL92 standard. The preferred embodiment supports ontologies specified in Web Ontology Language (OWL [OWL Web Ontology Language Reference, (available on the World Wide Web at http://www.w3.org/TR/owl-ref), specifically, OWL Lite and OWL DL) by extracting information from the OWL document and then storing this information in the schema.

[0026] The ontology-based operators and the indexing scheme employed in the preferred embodiment uses Oracle's Extensibility Framework as described by J. Srinivasan, R. Murthy, S. Sundara, N. Agarwal and S. DeFazio in "Extensible Indexing: A Framework for Integrating Domain-Specific Indexing into Oracle8i," Proceedings of the 16th International Conference on Data Engineering, pages 91-100, 2000. Specifically, the ONT_RELATED, ONT_DISTANCE, and ONT_PATH operators are implemented as user-defined operators and ONT_EXPAND is implemented as a table function. The operator implementation typically requires computing transitive closure, which is performed in Oracle SQL using queries with a CONNECT BY clause. Indexing is implemented as a user-defined indexing scheme. Although the ontology-based functions are described below in the context of an Oracle RDBMS, these functions can be supported in any RDBMS that supports the same basic capabilities provided by the Oracle RDBMS.

[0027] Before considering in detail how ontology-based matching and related functions are implemented, it will be useful to first consider how these operators might be used to provide the kind of semantic matching needed for the restaurant guide application noted in the background section above. To search the served_food database table for restaurants serving Latin American cuisine, the following SELECT statement might be used:

[0028] SELECT * FROM served_food WHERE ONT_RELATED (Cuisine, 'IS_A', 'Latin American', 'Cuisine_ontology') = 1;

[0029] The ONT_RELATED operator in the statement above evaluates two input terms,

a value from the Cuisine column in the table served_food and the string argument 'Latin

American'. The ONT_RELATED operator consults the specified ontology 'Cuisine_ontology' for the meaning of the two terms (shown graphically in Fig. 1) . If the operator determines that the two input terms are related by the input relationship type argument 'IS_A' by the ontology, it will return 1 (true), otherwise it returns 0 (false). The query thus identifies rows containing cuisines that are related to 'Latin American' based on the 'IS_A' relationship in the specified ontology and context, and would identify restaurants 2 and 14 which serve 'Mexican' and 'Portuguese' cuisine. The ONT_RELATED operator thus allows a user to introducing ontology- based semantic matching into SQL queries.

[0030] Optionally, as explained later in more detail, a user may want to get a measure for the rows identified by the ONT_RELATED operator. This can be achieved by using the ONT_DISTANCE ancillary operator. The ONT_DISTANCE operator gives a measure of how closely the terms are related by measuring the distance between the two terms. For example, the user may request that the results of the semantic matching query be sorted on this distance measure by submitting the following query:

[0031] SELECT * FROM served_food WHERE ONT_RELATED (cuisine, 'IS_A', 'Latin American', 'Cuisine_ontology', 123) = 1 ORDER BY ONT_DISTANCE (123);

[0032] In this query, the integer argument 123 in ONT_DISTANCE identifies the filtering operator expression (ONT_RELATED) that computes this ancillary value. Similarly, another ancillary operator named ONT_PATH may be used to compute the path measure value between the two terms. Ancillary operators are described by R. Murthy, S. Sundara, N. Agarwal, Y. Hu, T. Chorma and J. Srinivasan in "Supporting Ancillary Values from User Defined Functions in Oracle", In Proceedings of the 19th International Conference on Data Engineering, pages 151- 162, 2003.

[0033] In addition, a user may want to query an ontology independently (without involving user tables). The ONT_EXPAND operator described below can be used for this purpose.

[0034] Providing ontology-based semantic matching capability as part of SQL greatly facilitates developing ontology- driven database applications. Applications that can benefit include e-commerce (such as supply chain management, application integration, personalization, and auction). Also, applications that have to work with domain-specific

knowledge repositories (such as BioInformatics, Geographical Information Systems, and Healthcare Applications) can take advantage of this capability. These capabilities can be exploited to support semantic web applications such as web service discovery as well. A key requirement in these applications is to provide semantic matching between syntactically different terms or sometimes between syntactically same, but semantically different terms.

[0035] Support for ontology-based semantic matching is achieved by introducing the following extensions to existing database capabilities:

[0036] A. Two new SQL operators, ONT_RELATED and ONT_EXPAND are defined to model ontology based semantic matching operations. For queries involving ONT_RELATED operator, two ancillary SQL operators, ONT_DISTANCE and ONT_PATH, are defined that return distance and path respectively for the filtered rows.

[0037] B. A new indexing scheme ONT_INDEXTYPE is defined to speed up ontology-based semantic matching operations.

[0038] C. A system-defined ONTOLOGIES table is provided for storing ontologies.

[0039] In the description which follows: Section 2 presents an overview of the features which support ontology-based semantic matching operations; Section 3 discusses the implementation of the ontology-related functions by extending the existing capabilities of an Oracle RDBMS; and Section 4 illustrates the manner in which ontology-related functions may be used in several illustrative database applications.

[0040] 2. Supporting Ontology-based Semantic Matching in a Database System

[0041] 2.1 Overview

[0042] The principle ontology-related data structures and functions used in the preferred embodiment are illustrated in Fig. 2 and may be summarized as follows:

[0043] A top-level ONTOLOGIES table seen at 201 holds ontology data, which internally maps to a set of system- defined tables shown at 205.

[0044] Two operators are used for querying purposes. The ONT_EXPAND operator 211

can be used to query the ontology independently, whereas the ONT_RELATED operator 215 can be used to perform queries on one or more user tables 218 holding ontology terms whose meaning is specified by ontology data in the system defined tables 205. Optionally, a user can use ancillary operators ONT_DISTANCE and ONT_PATH operators in queries involving the ONT_RELATED operator 215 to get additional measures (distance and path) for the filtered rows extracted by the queries.

[0045] 2.2 RDBMS Schema for Storing Ontologies

[0046] An RDBMS schema has been created for storing ontologies specified in OWL. This RDBMS schema defines the following tables:

[0047] ***Ontologies***: Contains basic information about various ontologies, and includes the columns OntologyID, OntologyName, and Owner.

[0048] ***Terms***: Represents classes, individuals, and properties in the ontologies, and includes the column TermID, OntologyID, Term, and Type. A term is a lexical representation of a concept within an ontology. TermID value is generated to be unique across all ontologies. This allows representation of references to a term in a different ontology than the one that defines the term. Also, even an OntologyID is handled as a TermID which facilitates storing values for various properties (e.g., Annotation Properties) and other information that applies to an ontology itself. Note that, as a convention, any column in the above schema whose name is of the form "...ID..", would actually contain TermID values (like a foreign key).

[0049] ***Properties***: Contains information about the properties, and includes the columns OntologyID, PropertyID, DomainClassID, RangeClassID, and Characteristics. Domain and range of a property are represented with TermID values of the corresponding classes. Characteristics indicate which of the following properties are true for the property: symmetry, transitivity, functional, inverse functional.

[0050] ***Restrictions***: Contains information about property restrictions, and includes the columns OntologyID, NewClassID, PropertyID, MinCardinality, MaxCardinality, SomeValuesFrom, and AllValuesFrom. Restrictions on a property results in definition of a new class. This new class is not necessarily named (i.e., 'anonymous' class) in OWL. However, internally we create a new (system-defined) class for ease of representation.

[0051] ***Relationships***: Contains information about the relationship between two terms, and includes the OntologyID, TermID1, PropertyID, and TermID2.

[0052] ***PropertyValues***: Contains <Property, Value> pairs associated with the terms and includes the columns OntologyID, TermID, PropertyID, and Value. In order to handle values of different data types, some combinations of the following may be used: Define separate tables (or separate columns in the same table) for each of the frequently encountered types and use a generic self-describing type (ANYDATA in Oracle RDBMS) to handle any remaining types.

[0053] **System-defined Classes for Anonymous Classes:** We create internal (i.e., not visible to the user) or system-defined classes to handle OWL anonymous classes that arise in various situations such as Property Restrictions, enumerated types (used in DataRange), class definitions expressed as expression involving IntersectionOf, UnionOf, and ComplementOf.

[0054] **Bootstrap Ontology:** The first things that are loaded into the above schema are the basic concepts of OWL itself. In some sense this is like the bootstrap ontology. For example:

- Thing and Nothing are stored as Classes.
- subClassOf is stored as a transitive (meta) property that relates two classes.
- subPropertyOf is stored as a transitive (meta) property that relates two properties.
- disjointWith is stored as a symmetric (meta) property that relates two classes.
- SameAs is stored as a transitive and symmetric property that relates two individuals in Thing class.

[0055] Storing these OWL concepts as a bootstrap ontology facilitates inferencing. A simple example would be the following: If C1 is a subclassOf C2 and C2 is a subclassOf C3, then (by transitivity of subClassOf) C1 is a subclassOf C3. Note that the reflexive nature of subClassOf and SubPropertyOf is handled as a special case.

[0056] **Loading Ontologies:** An ontology is loaded into the database by using an API that takes as input an OWL document. Information from the OWL document is extracted and then stored into the system-defined tables in the RDBMS schema described above.

[0057] The Ontologies table stores some basic information about all the ontologies that are currently stored in the database. A portion (view) of this table is visible to the user.

[0058] 2.3 Modeling Ontology-based Semantic Matching

[0059] To support ontology-based semantic matching in RDBMS several new operators are defined.

[0060] **2.3.1 ONT_RELATED Operator.** This operator models the basic semantic matching operation. It determines if the two input terms are related with respect to the specified RelType relationship argument within an ontology. If they are related it returns 1, otherwise it returns 0.

[0061] ONT_RELATED (Term1, RelType, Term2, OntologyName) RETURNS INTEGER;

[0062] The RelType can specify a single ObjectProperty (for example, 'IS_A', 'EQV', etc.) or it can specify a combination of such properties by using AND, NOT, and OR operators (for example, 'IS_A OR EQV'). Note that both Term1 and Term2 need to be simple terms. If Term2 needs to be complex involving AND, OR, and NOT operators, user can issue query with individual terms and combine them with INTERSECT, UNION, and MINUS operators. See [0063] Section 2.3.4 for an example.

[0064] RelType specified as an expression involving OR and NOT operators (e.g., FatherOf OR MotherOf) is treated as a virtual relationship (in this case say AncestorOf) that is transitive by nature (also see Section 3.2.5).

[0065] **2.3.2 ONT_EXPAND Operator.** This operator is introduced to query an ontology independently. Similar to ONT_RELATED operator, the RelType can specify either a simple relationship or combination of them.

[0066] CREATE TYPE ONT_TermRelType AS OBJECT (

Term1Name VARCHAR(32),

PropertyName VARCHAR(32),

Term2Name VARCHAR(32),

TermDistance NUMBER,

TermPath VARCHAR(2000)

);

[0067] CREATE TYPE ONT_TermRelTableType AS TABLE OF ONT_TermRelType;

[0068] ONT_EXPAND (Term1, RelType, Term2, OntologyName

) RETURNS ONT_TermRelTableType;

[0069] Typically, non-NULL values for RelType and Term2 are specified as input and then the operator computes all the appropriate <Term1, RelType, Term2> tuples in the closure taking into account the characteristics (transitivity and symmetry) of the specified RelType. In addition, it also computes the relationship measures in terms of distance (TermDistance) and path (TermPath). For cases when a term is related to input term by multiple paths, one row per path is returned. It is also possible that ONT_EXPAND invocation may specify input values for any one or more of the three parameters or even none of the three parameters. In each of these cases, the appropriate set of <Term1, RelType, Term2> tuples is returned.

[0070] **2.3.3 ONT_DISTANCE and ONT_PATH Ancillary Operators.** These operators compute the distance and path measures respectively for the rows filtered using ONT_RELATED operator.

[0071] ONT_DISTANCE (NUMBER) RETURNS NUMBER;

[0072] ONT_PATH (NUMBER) RETURNS VARCHAR;

[0073] A single resulting row can be related in more than one way with the input term. For such cases, the above operators return the optimal measure, namely smallest distance or shortest path. For computing all the matches, the following two operators are provided:

[0074] ONT_DISTANCE_ALL (NUMBER)

RETURNS TABLE OF NUMBER;

[0075] ONT_PATH_ALL (NUMBER)

RETURNS TABLE OF VARCHAR;

[0076] **2.3.4 A Restaurant Guide Example.** Consider a restaurant guide application that maintains type of cuisine served at various restaurants. It has two tables, 1) restaurants containing restaurant information, and 2) servedfood containing the types of cuisine served at restaurants.

[0077] The restaurant guide application takes as input a type of cuisine and returns the list of restaurants serving that cuisine. Obviously, applications would like to take advantage of an available cuisine ontology to provide better match for the user queries. The cuisine ontology describes the relationships between various types of cuisines as shown earlier in Figure 1.

[0078] Thus, if a user is interested in restaurants that serve cuisine of type 'Latin American', the database application can generate the following query:

[0079]
```
SELECT r.name, r.address
FROM served_food sf, restaurant r
WHERE r.id = sf.r_id AND  ONT_RELATED(sf.cuisine, 'IS_A OR EQV',
        'Latin American',
        'Cuisine_ontology')=1;
```

[0080] To query on 'Latin American' AND 'Western' the application program can obtain rows for each and use the SQL INTERSECT operation to compute the result.

[0081] Also, the application can exploit the full SQL expressive power when using ONT_RELATED operator. For example, it can easily combine the above query results with those restaurants that have lower price range.

[0082]
```
SELECT r.name FROM served_food sf, restaurant r
    WHERE r.id = sf.r_id AND ONT_RELATED(sf.cuisine, 'IS_A OR EQV',
        'Latin American',
```

'Cuisine_ontology')=1 AND r.price_range = '$';

[0083] **2.3.5 Discussion.** Note that the queries in section 2.3.4 can also be issued using the ONT_EXPAND operator. For example, the first query in that section can alternatively be expressed using ONT_EXPAND as follows:

[0084] SELECT r.name, r.address FROM served_food sf, restaurant r

WHERE r.id = sf.r_id AND sf.cuisine IN

(SELECT Term1Name from TABLE(ONT_EXPAND(NULL, 'IS_A OR EQV',

'Latin American', 'Cuisine_ontology')));

[0085] The ONT_RELATED operator is provided in addition to ONT_EXPAND operator for the following reasons:

- The ONT_RELATED operator provides a more natural way of expressing semantic matching operations on column holding ontology terms; and
- It allows use of an index created on column holding ontology terms to speed up the query execution by taking column data into account.

[0086] 2.4 Inferencing

[0087] Inferencing rules employing the symmetry and transitivity characteristics of properties are used to infer new relationships. This kind of inferencing can be achieved through the use of the operators defined above (see Section 3.2 for details). Note that our support for inferencing is restricted to OWL Lite and OWL DL, both of which are decidable.

[0088] To support more complete inferencing using the above operators, an initial phase of inferencing is done after ontology is loaded, and the results of this inferencing are stored persistently and used in subsequent inferencing. Several examples of the initial inferencing follow.

[0089] All subPropertyOf relationships are derived and stored during this phase. The transitive aspects of sameAs (e.g., sameAs(x,y) AND sameAs(y,z) IMPLIES sameAs(x,z)) can be handled by the operators defined above. However, the more complex rules which

imply sameAs (e.g., p is a functional property AND p(a,x) AND p(b,y) AND sameAs(a,b) IMPLIES sameAs(x,y)) are best handled outside of the operator implementation. We expect these complex sameAs inferences to be done during the initial inferencing phase. To provide the semantics of sameAs during closure computation, the operators will treat an individual 'I' as 'I OR J' for all J where sameAs(I, J).

[0090] Furthermore, we are introducing an internal relationship that will be useful for inferencing over complex subPropertyOf and inverseOf interactions: SubPropertyOfInverseOf(f,g) iff f(x,y) IMPLIES g(y,x). Again, we expect that the rules that introduce SubPropertyOfInverseOf (spiOf, for short) into an ontology (e.g., inverseOf(f,g) IMPLIES spiOf(f,g) AND spiOf(g,f)) will be handled during the initial inferencing phase. However, the transitive aspects of spiOf can be handled as a special case within our operators, according to the following rules:

- subPropertyOf(f,g) AND spiOf(g,h) IMPLIES spiOf(f,h)
  (Proof: f(x,y) IMPLIES g(x,y) IMPLIES h(y,x))
- spiOf(f,g) AND subPropertyOf(g,h) IMPLIES spiOf(f,h)
  (Proof: f(x,y) IMPLIES g(y,x) IMPLIES h(y,x))
- spiOf(f,g) AND spiOf(g,h) IMPLIES SubPropertyOf(f,h)
  (Proof: f(x,y) IMPLIES g(y,x) IMPLIES h(x,y))

[0091] Given the expansion of subPropertyOf and spiOf, we can find all relationship tuples for a given property, including those that are implied through sub-properties and inverse-properties. Consider the following example where non-transitive properties are used for clarity, which expands ParentOf. In this case, if we have subPropertyOf(MotherOf, ParentOf) and inverseOf(MotherOf, hasMother), we will get spiOf(hasMother, ParentOf) based on result of the initial inferencing phase and the above rules. Then hasMother(child, mother) will be sufficient to yield ParentOf(mother, child) in the result set:

[0092]
```
SELECT r.termID1, 'ParentOf', r.termID2 FROM relationships r, terms t
 WHERE r.propertyID = t.termID  AND t.term IN
  (select term1Name FROM
  TABLE(ONT_EXPAND(NULL,
    'subPropertyOf',
```

```
        'ParentOf',
        'Family_ontology')))
UNION
SELECT r.termID2, 'ParentOf', r.termID1
FROM relationships r, terms t
WHERE r.propertyID = t.termID
AND t.term IN
(select term1Name FROM
  TABLE(ONT_EXPAND(NULL,
    'spiOf',
    'ParentOf',
    'Family_ontology')))
```

[0093] **3. Implementation of Ontology Related Functionality on Oracle RDBMS**

[0094] This section describes how the ontology-related functionality is implemented on top of Oracle RDBMS

[0095] 3.1 Operators

[0096] The ONT_RELATED operator is defined as a *primary* user-defined operator, with ONT_DISTANCE and ONT_PATH as its *ancillary* operators. The primary operator computes the ancillary value as part of its processing [97]. In this case, ONT_RELATED operator computes the relationship. If ancillary values (the distance and path measure) are required, it computes them as well.

[0098] Note that the user-defined operator mechanism in Oracle allows for sharing state across multiple invocations. Thus, the implementation of the ONT_RELATED operator involves building and compiling an SQL query with CONNECT BY clause (as described in Section 3.2) during its first invocation. Each subsequent invocations of the operator simply

uses the previously compiled SQL cursor, binds it with the new input value, and executes it to obtain the result.

[0099] The ONT_EXPAND operator is defined as a table function as it returns a table of rows, which by default includes the path and distance measures.

[0100] 3.2 Basic Algorithm

[0101] Basic processing for both ONT_RELATED and ONT_EXPAND involves computing transitive closure, namely, traversal of a tree structure by following relationship links given a starting node. Also, as part of transitive closure computation, we need to track the distance and path information for each pair formed by starting node and target node reached via the relationship links.

[0102] Oracle supports transitive closure queries with CONNECT BY clause as follows:

[0103] SELECT ... FROM ... START WITH <condition>

CONNECT BY <condition>;

[0104] The starting node is selected based on the condition given in START WITH clause, and then nodes are traversed based on the condition given in CONNECT BY clause. The parent node is referred to by the PRIOR operator. For computation of distance and path, the Oracle-provided LEVEL psuedo-column and SYS_CONNECT_BY_PATH function are respectively used in the select list of a query with CONNECT BY clause.

[0105] Note that in the system-defined *Relationships* table, a row represents 'TermID1 is related to TermID2 via PropertyID relationship.' For example, if 'A IS_A B', it is represented as the row <1, A, IS_A , B> assuming that the ontologyID is 1.

[0106] Note that any cycles encountered during the closure computation will be handled by the CONNECT BY NOCYCLE query implementation available in Oracle 10g (not explicitly shown in the examples below).

[0107] For simplicity, we use a slightly different definition for the relationships table where term names are stored instead of termIDs. In this case, the Relationships table has the columns (OntologyName, Term1, Relation, Term2, ...).

[0108] To illustrate the processing, we use the restaurant guide example. The data in the restaurant and served_food tables is shown below:

[0109] **restaurant**

Id	Name	price_range	.....
1	Mac	$	
2	Chilis	$$	
3	Anthonys	$$$	
4	BK	$	
5	Uno	$$	
6	Wendys	$	
7	Dabin	$$	
8	Cheers	$$	
9	KFC	$	
10	Sizzlers	$$	
11	Rio	$$	
12	Maharaj	$$	
13	Dragon	$$	
14	Niva	$$	

[0110] **served_food**

R_id	cuisine
1	American
2	Mexican
2	American
3	American
4	American

5	American
5	Italian
6	American
7	Korean
7	Japanese
8	American
9	American
10	American
11	Brazilian
12	Mexican
12	Indian
13	Chinese
14	Portuguese

[0111] **3.2.1 Handling Simple Terms.** Consider a query that has simple relation types, i.e., no AND, OR, NOT operators. The first query given in Section 2.3.4 can be converted as follows:

[0112] Original Query:

[0113] SELECT r.name, r.address FROM served_food sf, restaurant r

WHERE r.id = sf.r_id AND

ONT_RELATED(sf.cuisine, 'IS_A', 'Latin American', Cuisine_ontology')=1;

[0114] Transformed Query:

[0115] SELECT r.name, r.address

FROM served_food sf,restaurant r

```
WHERE r.id = sf.r_id AND
    sf.cuisine IN (SELECT term1 FROM relationships
START WITH
    term2 = 'Latin American' AND relation = 'IS_A'
    CONNECT BY
PRIOR term1 = term2 AND relation = 'IS_A');
```

[0116] The text in boldface above is the portion that has been converted. Basically, the third argument is translated into START WITH clause and the second argument into CONNECT BY clause. The result for this query is as follows:

[0117] Query Result

NAME	ADDRESS
Chilis	
.....	
Maharaj	
.....	
Niva	.....

[0118] **3.2.2 Handling OR Operator.** Consider a case where 'Brazilian' cuisine was not originally included in the ontology and is now inserted under the 'South American' cuisine. Also, to put 'South American' cuisine in the same category as 'Latin American' cuisine, the transitive and symmetric 'EQV' relationship is used as shown in Figure 3:

[0119] Now, to get 'Latin American' cuisine, disjunctive conditions should be used to traverse both relationship links, that is, 'IS_A' and 'EQV'. Such disjunctive conditions can be directly specified in the START WITH and CONNECT BY clauses.

[0120] Original Query:

[0121] SELECT r.name, r.address FROM served_food sf, restaurant r

WHERE r.id = sf.r_id AND

ONT_RELATED(sf.cuisine,

'IS_A OR EQV',

'Latin American',

'Cuisine_ontology')=1;

[0122] Transformed Query:

[0123] The only differences from the transformed query of the previous example is that the relationships table:

FROM relationships

is replaced by a sub-query to introduce the implicit symmetric edges into the query:

FROM (SELECT term1, relation, term2

FROM relationships

UNION

SELECT term2, relation, term1

FROM relationships

WHERE relation = 'EQV')

and the occurrence of the following predicate in START WITH and CONNECT BY clauses

relation = 'IS_A'

is replaced with the following predicate:

(relation = 'IS_A' OR relation = 'EQV')

[0124] **3.2.3 Handling AND operator.** Conjunctive conditions between transitive relationship types can be handled by independently computing the transitive closure for each relationship type and then applying set INTERSECT on the resulting sets. For each node in the intersection, a path exists from the start node to this node for each relationship type and hence this is sufficient.

[0125] Let us consider another relationship between cuisines, which identifies the spiciest cuisine using the term MOST_SPICY. The ontology can now contain information such as 'South Asian cuisine is MOST_SPICY Asian cuisine' and 'Indian cuisine is MOST_SPICY South Asian cuisine,' etc.

[0126] To find very spicy cuisine from the ontology, user can issue a query using conjunctive conditions in the relationships as follows:

[0127] Original Query: Find a restaurant that serves very spicy Asian cuisine.

[0128]
```
SELECT r.name FROM served_food sf, restaurant r
WHERE r.id = sf.r_id AND
ONT_RELATED(sf.cuisine,
    'IS_A AND MOST_SPICY'
    'Asian',
    'Cuisine_ontology') = 1;
```

[0129] Transformed query:

[0130]
```
SELECT r.name FROM served_food sf, restaurant r
WHERE r.id = sf.r_id AND
 sf.cuisine IN
(
SELECT term1 FROM relationships
START WITH
term2 = 'Asian' AND
        relation = 'IS_A'
        CONNECT BY
```

```
PRIOR term1 = term2 AND
relation = 'IS_A'
INTERSECT
SELECT term1 FROM relationships
START WITH
        term2 = 'Asian' AND
        relation = 'MOST_SPICY'
                CONNECT BY
PRIOR term1 = term2 AND
relation ='MOST_SPICY');
```

[0131] **3.2.4 Handling NOT operator.** A NOT operator specifies which relationships to exclude when finding transitive closure. Therefore, given the start node all relationships except ones specified in NOT operator will be traversed. NOT operators can be directly specified in the START WITH and CONNECT BY clauses.

[0132] Original Query: Find all Latin American cuisine, excluding 'EQV' relationship types.

[0133]
```
SELECT r.name FROM served_food sf, restaurant r
WHERE r.id = sf.r_id AND
    ONT_RELATED(sf.cuisine,
            'NOT EQV',
            'Latin American',
            'Cuisine_ontology')=1;
```

[0134] Transformed Query: Only difference from the transformed query of the example in Section 3.2.1 is that the occurrence of the following predicate in START WITH and CONNECT BY clauses

relation = 'IS_A'

is replaced with the following predicate:

relation != 'EQV'

[0135] Note that if a user wants to retrieve all cuisines except Latin American cuisine, then the query can be formulated using the operator ONT_RELATED returning 0 as follows:

```
.....
ONT_RELATED(sf.cuisine,
    'IS_A',
    'Latin American',
'Cuisine_ontology')=0;
```

[0136] **3.2.5 Handling Combination of OR, AND, and NOT.** OR and NOT operators are directly specified in the CONNECT BY clause and AND operators are handled by INTERSECT. All conditions are rewritten as conjunctive conditions. For example, 'A OR (B AND C)' will be converted into '(A OR B) AND (A OR C).' Then, '(A OR B)' and '(A OR C)' are specified in the CONNECT BY clause in separate queries that can be combined with INTERSECT operator.

[0137] 4. Ontology-driven Database Applications

[0138] This section illustrates the usage of the ontology-related semantic matching operations by considering several database applications.

[0139] 4.1 A Homeland Security Application

[0140] An intelligence analyst for homeland security would be very much interested in an activity that might be related to terrorism. From the example pattern given in [20], where a person rents a truck and another person buys fertilizer, and they reside in the same house, we can formulate an SQL query using ONT_RELATED operator for a given activity table:

[0142] Table 5

Person_name	Address	Activity	Object
John Buck	Addr1	Rent	Ford F-150
Jane Doe	Addr1	Buy	Ammonium Nitrate
...	...	...	...

[0143]
```
SELECT *
  FROM ACTIVITY x, ACTIVITY y
  WHERE
    x.Activity = 'Rent' AND
    y.Activity = 'Buy' AND
    ONT_RELATED(x.object,'IS_A','Truck',
        'vehicle_ontology') = 1 AND
    ONT_RELATED(y.object,'IS_A','Fertilizer',
        'chemical_ontology')=1 AND
    x.Address = y.Address;
```

[0144] By referring to more than one ontology we can analyze suspicious activities involving a combination of different actions.

[0145] 4.2 A Supply Chain Application

[0146] A supply chain where thousands of products and services are exchanged has a major issue of standardizations of purchase order, bill of material, catalogs, etc. There could be name, language, currency, and unit differences to resolve to name a few. Standardization efforts such as RosettaNet [22], UNSPSC [23] for product category and ebXML [24] to standardize business processes and products are not enough to meet individual vendor/customer needs to resolve semantic differences. Typically, these conflicts have been resolved using some form of mapping mechanisms [6].

[0151] These conflicts can be resolved by issuing an SQL query with ONT_RELATED operator using ontologies. Even individually developed ontologies may need the ontology mappings to communicate each other. We can apply ONT_RELATED and ONT_EXPAND operators to the mapping ontology to resolve the conflicts as well.

[0152] 4.3 A Life Science Application

[0153] Life Science domain applications have been using ontologies for representing knowledge as well as the basis of information integration of several heterogeneous sources of life science web databases. Let us consider Gene Ontology (GO)[12], which is primarily used to represent the current knowledge in this domain. It allows user to query using SQL. One sample query is '*Fetching every term that is a transmembrane receptor'*. The GO Graph is stored using the 'term' (=node) and 'term2term' (=arc) tables. It also maintains a table called 'graph_path', which stores all paths between a term and all its ancestors. In GO database the following SQL query can be issued for this purpose:

[0155]
```
SELECT
  rchild.*
FROM
  term AS rchild, term AS ancestor,
  graph_path
WHERE
  graph_path.term2_id = rchild.id and
  graph_path.term1_id = ancestor.id and
  ancestor.name = 'transmembrane receptor';
```

[0156] The following query can be used if the data is stored in Oracle RDBMS:

[0157]
```
SELECT * FROM TABLE
  (     ONT_EXPAND (NULL, 'IS_A',
```

'transmembrane receptor', 'gene_ontology'));

[0158] The key difference is that the GO database exposes the underlying schema and requires users to formulate queries against those tables, whereas the operator approach attempts to simplify the specification.

[0159] 4.4 A Web Service Application

[0160] Similarly, web service applications can also utilize the ONT_RELATED operator to match two different terms semantically. Consider the web-service matching example described in [8], where user is looking to purchase a sedan. The user requests a web service using the term 'Sedan'. The vehicle ontology contains 'Sedan', 'SUV', and 'Station Wagon' as subclasses of the term 'Car'. The web service can alert the user by using the query with the ONT_RELATED operator as follows:

```
.....
ONT_RELATED(user_request,
    'IS_A',
    'Car',
        'Vehicle_Ontology')
= 1;
```

[0162] The degree of match between terms, i.e. how closely the terms are related, as described in [8], can be handled using ONT_DISTANCE and ONT_PATH operators.

[0164] 5. Conclusions

[0165] With the increasing importance of ontologies in business database application areas, it is important that ontologies are stored in databases for efficient maintenance, sharing, and scalability. Equally important is the capability for ontology-based semantic matching in SQL for performance and ease of application development.

[0166] The specific embodiment of the invention which has been described addresses these issues by allowing OWL Lite and OWL DL based ontologies to be stored in Oracle RDBMS and by providing a set of SQL operators for ontology-based semantic matching.

[0167] It is to be understood that the specific examples described above are merely illustrative applications of the principles of the invention. Numerous modifications may be made to the methods, data structures and SQL statements that have been presented without departing from the true spirit and scope of the invention.

What is claimed is:

1. The method for processing data stored in a relational database comprising, in combination, the steps of:

storing ontology data that specifies terms and relationships between pairs of said terms,

forming a database query that includes a semantic matching operator that identifies said ontology data and a further specifies a stated relationship between two input terms, and

executing said query to invoke said semantic matching operator to determine if said two input terms are related by the stated relationship by consulting said ontology data.

2. The method for processing data stored in a relational database as set forth in claim 1 wherein said ontology data specifies relationships of different types between said pairs of terms and wherein said semantic matching operator further specifies the type of relationship that must exist between said selected data and said term value.

3. The method for processing data stored in a relational database as set forth in claim 2 wherein said ontology data is identified by an ontology name value and wherein said semantic matching operator specifies said ontology data using said ontology name value.

4. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database where said data values have relationships to argument values that are defined in ontology data, said method comprising the step of:

forming and executing an SQL query statement that includes the identification of a semantic matching operator that is invoked to identify said target data values by consulting said ontology data to determine whether said target data values are related to said argument values.

5. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 4 wherein said

identification of a semantic matching operator is a user-defined operator that extends the pre-existing SQL syntax employed by said relational database.

6. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 4 wherein said SQL query statement further includes the identification of said one or more selected columns.

7. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 6 wherein said semantic matching operator is invoked to match data values in said one or more selected columns one with said argument values by consulting said ontology data.

8. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 4 wherein said ontology data specifies relationships of different types between said data values and said argument values and wherein said semantic matching operator further specifies the type of relationship that must exist between said selected data and said term value.

9. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 8 wherein at least some of said relationships of different types are transitive and wherein said semantic matching operator is invoked to search transitive pathways connected by said transitive relationships.

10. The method of identifying target data values stored in one or more selected columns of selected tables in a relational database as set forth in claim 9 wherein said semantic matching operator further computes a measure of how closely said data values are related to said argument values by measuring said transitive pathways.

11. The method of performing an ontology-based matching operation during the execution of a relational database query defined by a structured query language statement, said method comprising, in combination, the steps of:

creating and storing in said relational database a semantic matching operator that determines if two input terms are related by a specified type of relationship as defined by specified ontology data, and

constructing and executing a query expressed in the structured query language that includes an identification of said semantic matching operator.

12. The method of performing an ontology-based matching operation as set forth in claim 11 wherein said ontology data specifies relationships of different types between said pairs of terms and wherein said identification of said semantic matching operator further specifies the type of relationship that must exist between said two input terms.

13. The method of performing an ontology-based matching operation as set forth in claim 12 wherein at least some of said different types of relationships are properties each of which specifies a relationship between one term one or more other terms and wherein said semantic matching operator specifies one of said properties.

14. The method for performing an ontology-based matching operation as set forth in claim 13 wherein said ontology data further specifies that at least one of said property is a subproperty of another of said properties and wherein said semantic matching operator specifies properties that are subproperties of an identified property.

15. The method for performing an ontology-based matching operation as set forth in claim 14 wherein said properties define transitive relationships and wherein said semantic matching operator when executed infers new relationships based on said transitive relationships.

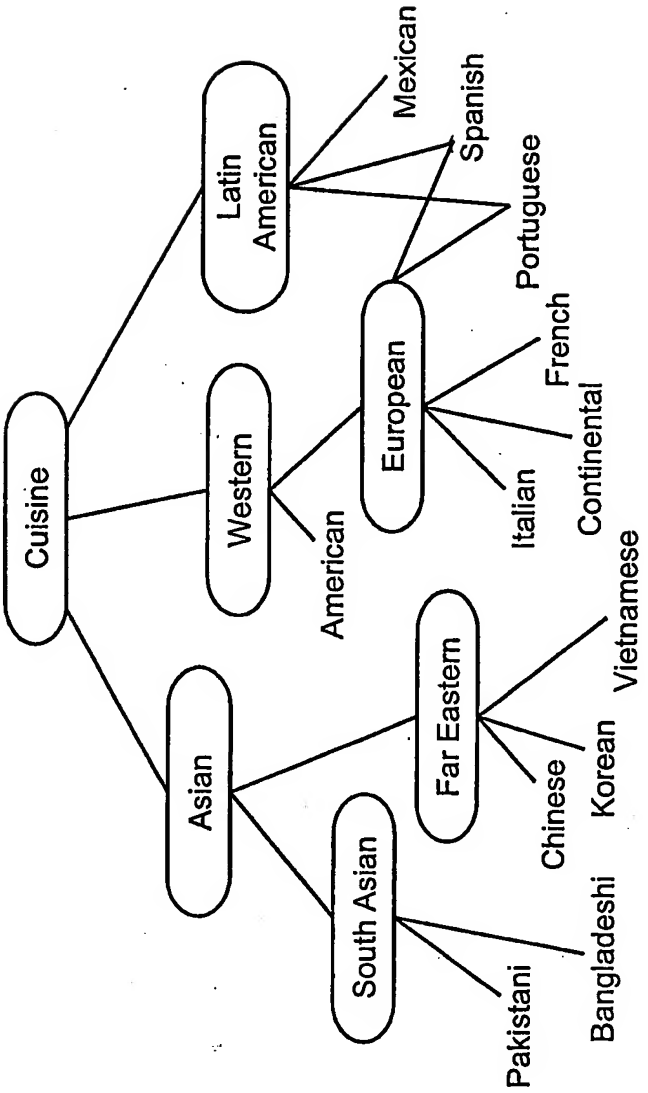

Cuisine
Latin American
Mexican
Spanish
Portuguese
Western
European
French
Continental
Italian
American
Asian
Far Eastern
Vietnamese
Korean
Chinese
South Asian
Bangladeshi
Pakistani

Fig. 1

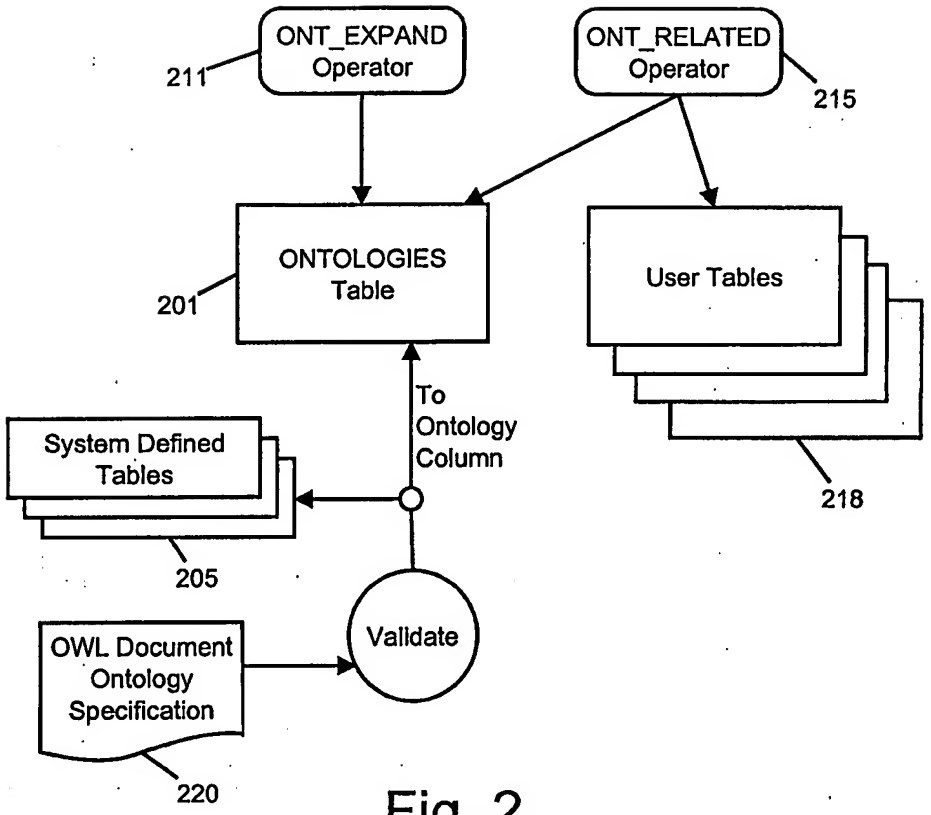

ONT_EXPAND Operator
211
ONT_RELATED Operator
215
ONTOLOGIES Table
201
User Tables
218
To Ontology Column
System Defined Tables
205
Validate
OWL Document Ontology Specification
220

Fig. 2

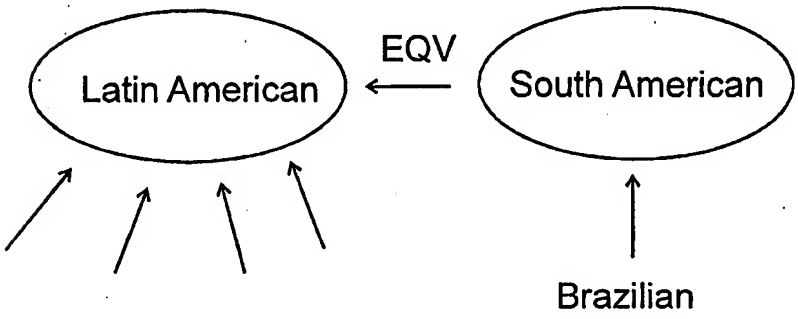

Latin American
EQV
South American
Brazilian

Fig. 3

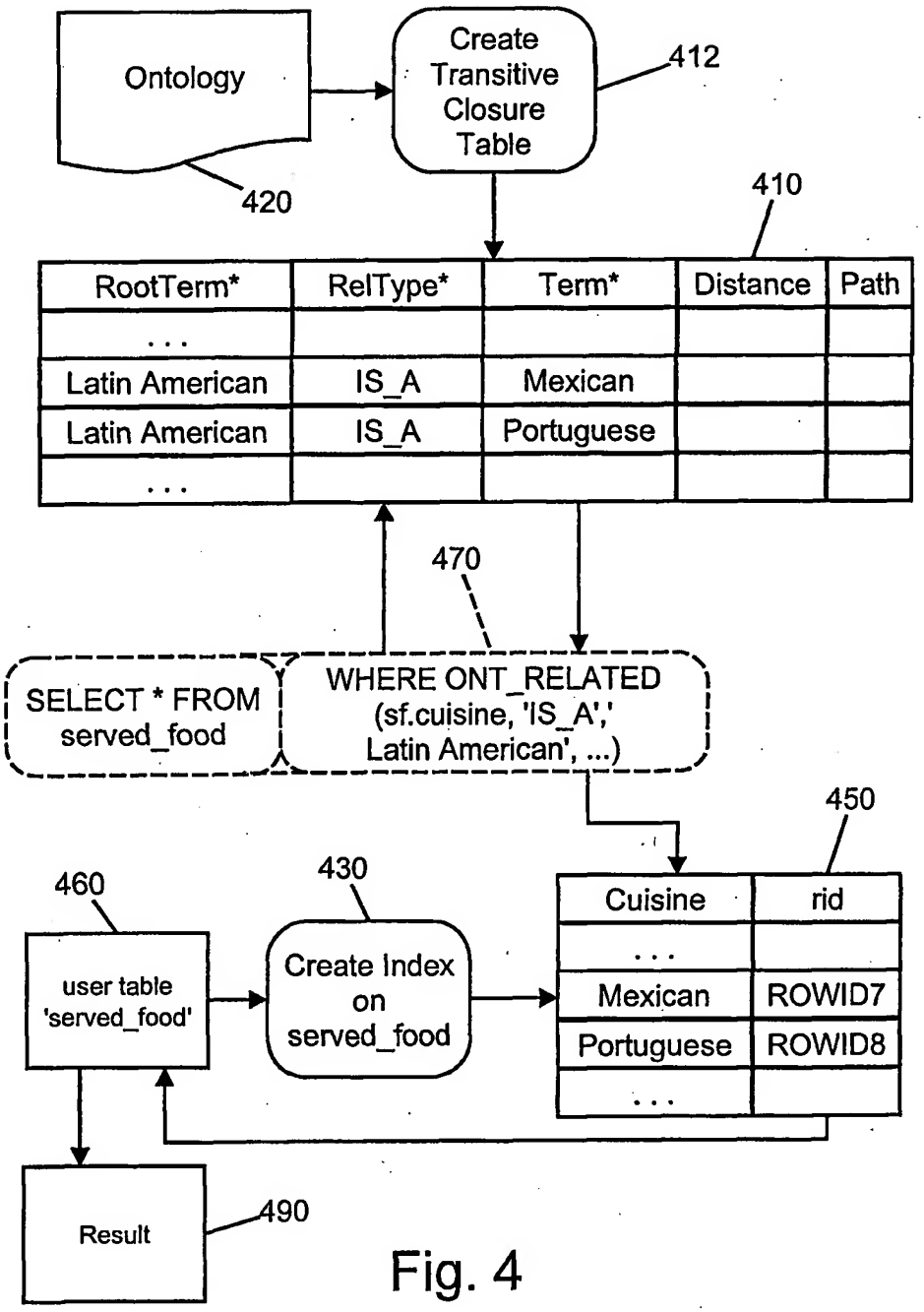

Ontology
Create Transitive Closure Table
412
420
410
RootTerm*
RelType*
Term*
Distance
Path
Latin American
IS_A
Mexican
Latin American
IS_A
Portuguese
470
SELECT * FROM served_food
WHERE ONT_RELATED (sf.cuisine, 'IS_A',' Latin American', ...)
450
460
430
user table 'served_food'
Create Index on served_food
Cuisine
rid
Mexican
ROWID7
Portuguese
ROWID8
Result
490

Fig. 4

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2005/025975

A. CLASSIFICATION OF SUBJECT MATTER
G06F17/30

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practical, search terms used)
EPO-Internal, BIOSIS, EMBASE, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	KOEHLER JACOB ET AL: "The semantic metadatabase (SEMEDA): Ontology based integration of federated molecular biological data sources."'Online! pages 1-10, XP002359704 Retrieved from the Internet: URL:http://web.archive.org/web/20030123222238/http://www.bioinfo.de/isb/2002/02/0021/> 'retrieved on 2003-01-23! abstract page 2 - page 4 -/--	1-15

[X] Further documents are listed in the continuation of box C.

[X] Patent family members are listed in annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"E" earlier document but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art.

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 December 2005	10/01/2006

Name and mailing address of the ISA	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Tx. 31 651 epo nl, Fax: (+31-70) 340-3016	San-Bento Furtado, P

Form PCT/ISA/210 (second sheet) (January 2004)

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
	& DATABASE BIOSIS 'Online! BIOSCIENCES INFORMATION SERVICE, PHILADELPHIA, PA, US; 2002, KOEHLER JACOB ET AL: "The semantic metadatabase (SEMEDA): Ontology based integration of federated molecular biological data sources." Database accession no. PREV200300124429 abstract & IN SILICO BIOLOGY, vol. 2, no. 3, 2002, pages 219-231, ISSN: 1386-6338	
X	KOHLER J ET AL: "SEMEDA: Ontology based semantic integration of biological databases" BIOINFORMATICS, vol. 19, no. 18, 12 December 2003 (2003-12-12), pages 2420-2427, XP002359982 UNITED KINGDOM ISSN: 1367-4803 page 2421 - page 2423 & DATABASE EMBASE 'Online! ELSEVIER SCIENCE PUBLISHERS, AMSTERDAM, NL; 12 December 2003 (2003-12-12), Database accession no. EMB-2004004501 abstract	1-15
X	WO 2004/053633 A (DECODE GENETICS EHF) 24 June 2004 (2004-06-24) abstract page 9	1-15
X	US 6 640 231 B1 (ANDERSEN WILLIAM A ET AL) 28 October 2003 (2003-10-28) abstract column 13; figure 1	1-15
X	KOHLER J ET AL: "Logical and semantic database integration" BIO-INFORMATICS AND BIOMEDICAL ENGINEERING, 2000. PROCEEDINGS. IEEE INTERNATIONAL SYMPOSIUM ON 8-10 NOVEMBER 2000, PISCATAWAY, NJ, USA,IEEE, 8 November 2000 (2000-11-08), pages 77-80, XP010526454 ISBN: 0-7695-0862-6 the whole document	1-15

-/--

INTERNATIONAL SEARCH REPORT

International Application No
PCT/US2005/025975

C.(Continuation) DOCUMENTS CONSIDERED TO BE RELEVANT

Category *	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	BONATTI P ET AL: "An ontology-extended relational algebra" INFORMATION REUSE AND INTEGRATION, 2003. IRI 2003. IEEE INTERNATIONAL CONFERENCE ON OCT. 27-29, 2003, PISCATAWAY, NJ, USA,IEEE, 27 October 2003 (2003-10-27), pages 192-199, XP010673718 ISBN: 0-7803-8242-0 abstract page 192, left-hand column	1-15
A	US 2004/030687 A1 (HIDAKA YUFUKO ET AL) 12 February 2004 (2004-02-12) abstract page 2 - page 3	1-15
A	W3C: "OWL Web Ontology Language Overview"'Online! 10 February 2004 (2004-02-10), pages 1-15, XP002359705 Retrieved from the Internet: URL:http://web.archive.org/web/20040225212232/http://www.w3.org/TR/2004/REC-owl-features-20040210/> 'retrieved on 2004-02-25! abstract page 6 - page 7	1-15

5

INTERNATIONAL SEARCH REPORT

Information on patent family members

International Application No
PCT/US2005/025975

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 2004053633 A	24-06-2004	AU 2003283042 A1	30-06-2004
US 6640231 B1	28-10-2003	AU 8898601 A WO 0231680 A1	22-04-2002 18-04-2002
US 2004030687 A1	12-02-2004	JP 2004062446 A	26-02-2004